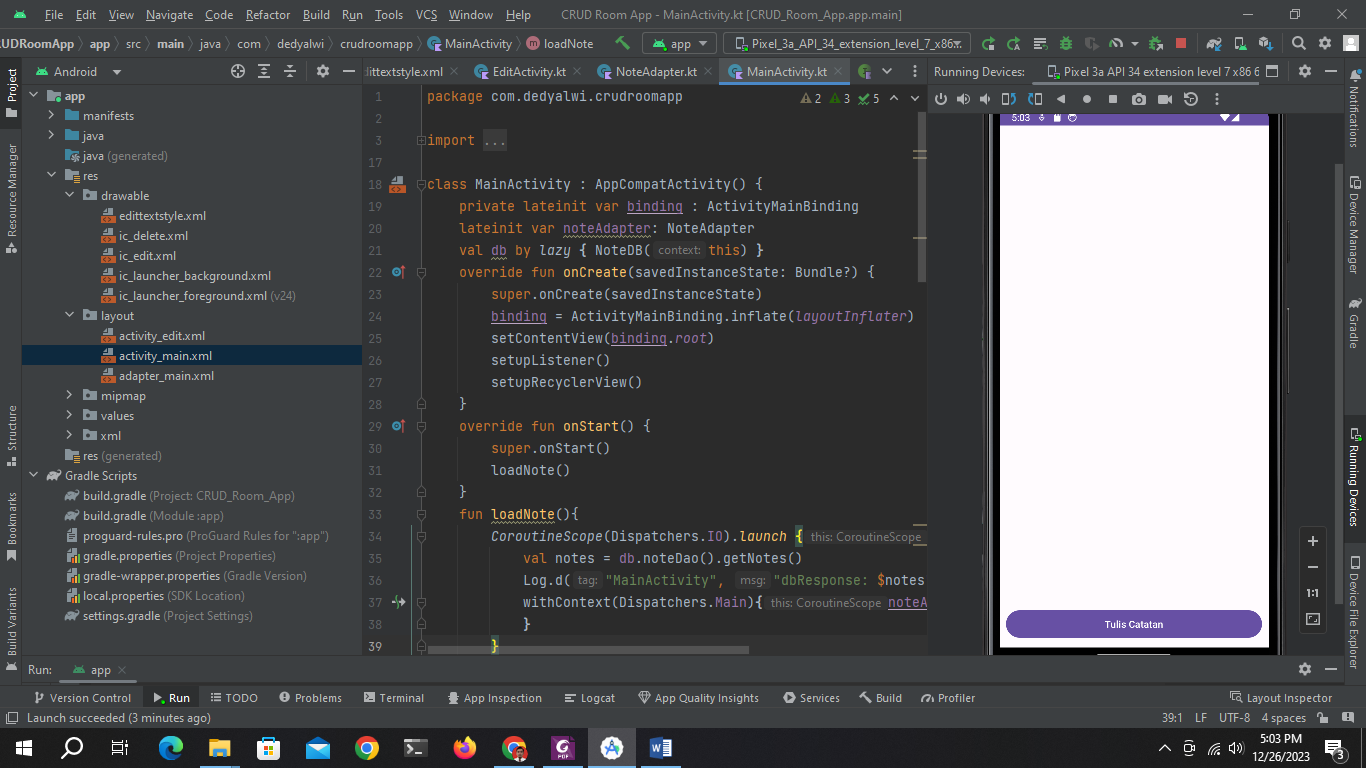
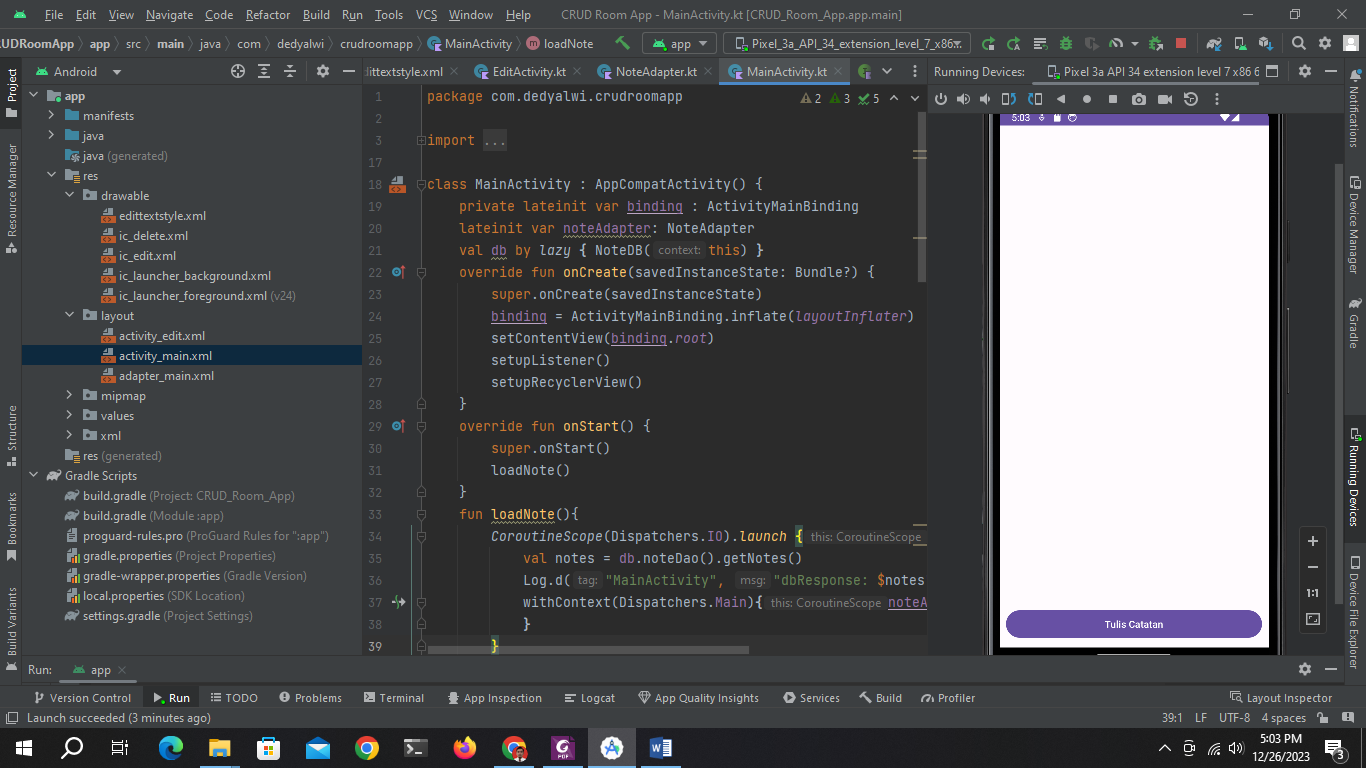
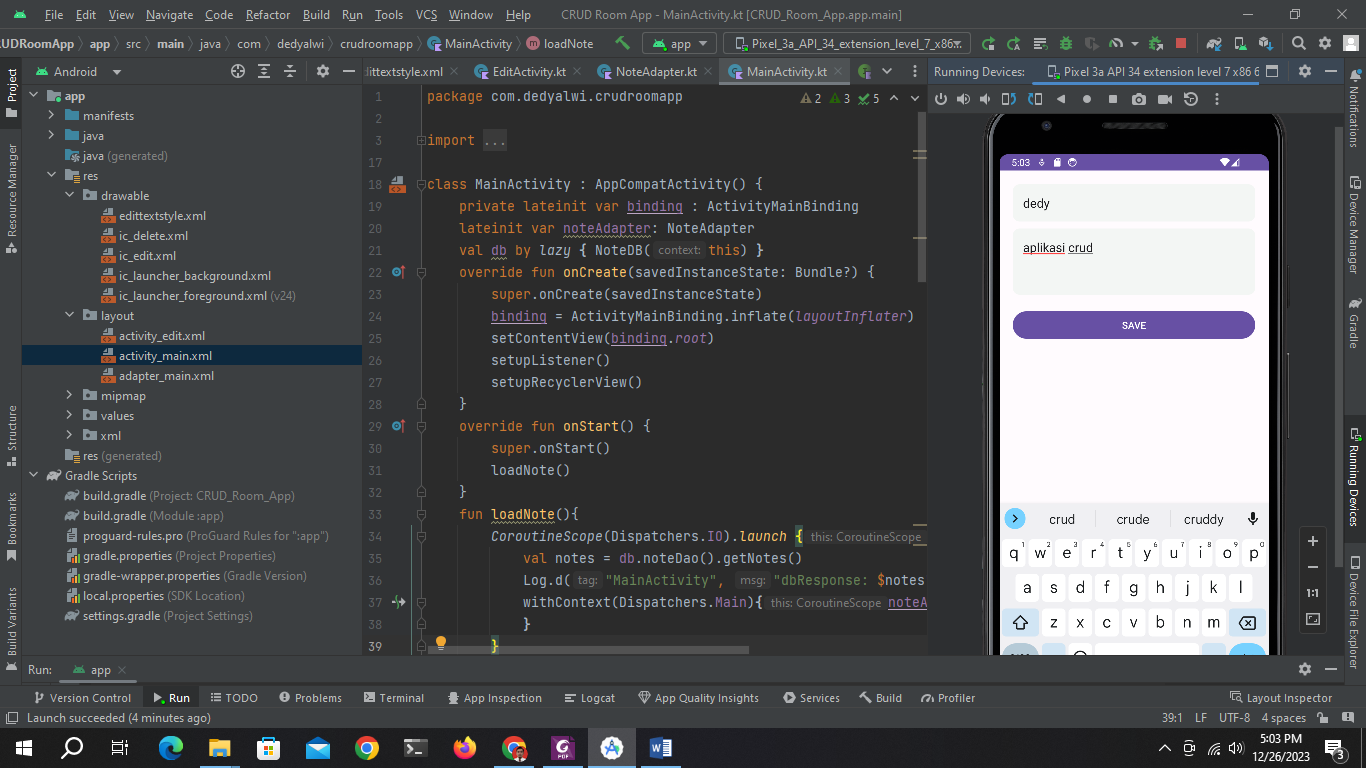
Nama : Dedy Alwi

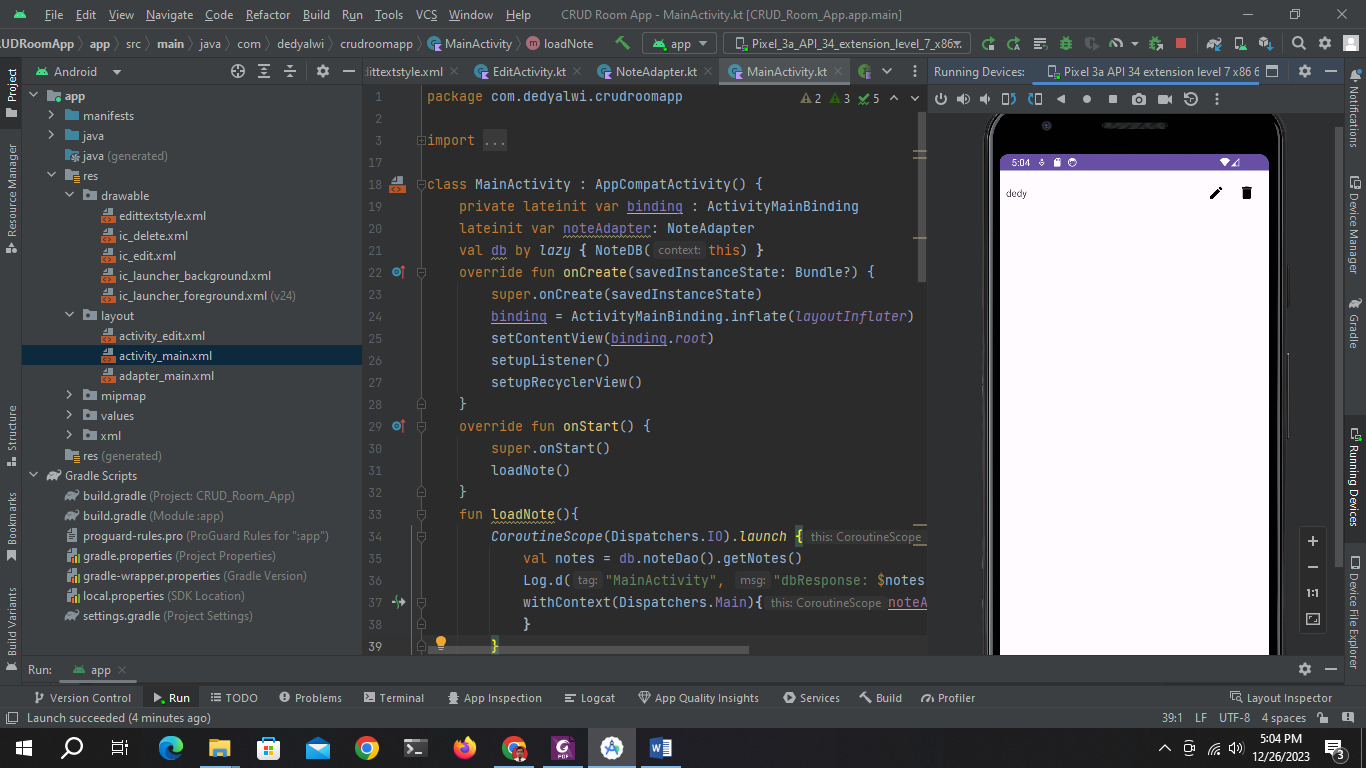
NIM : 23510015

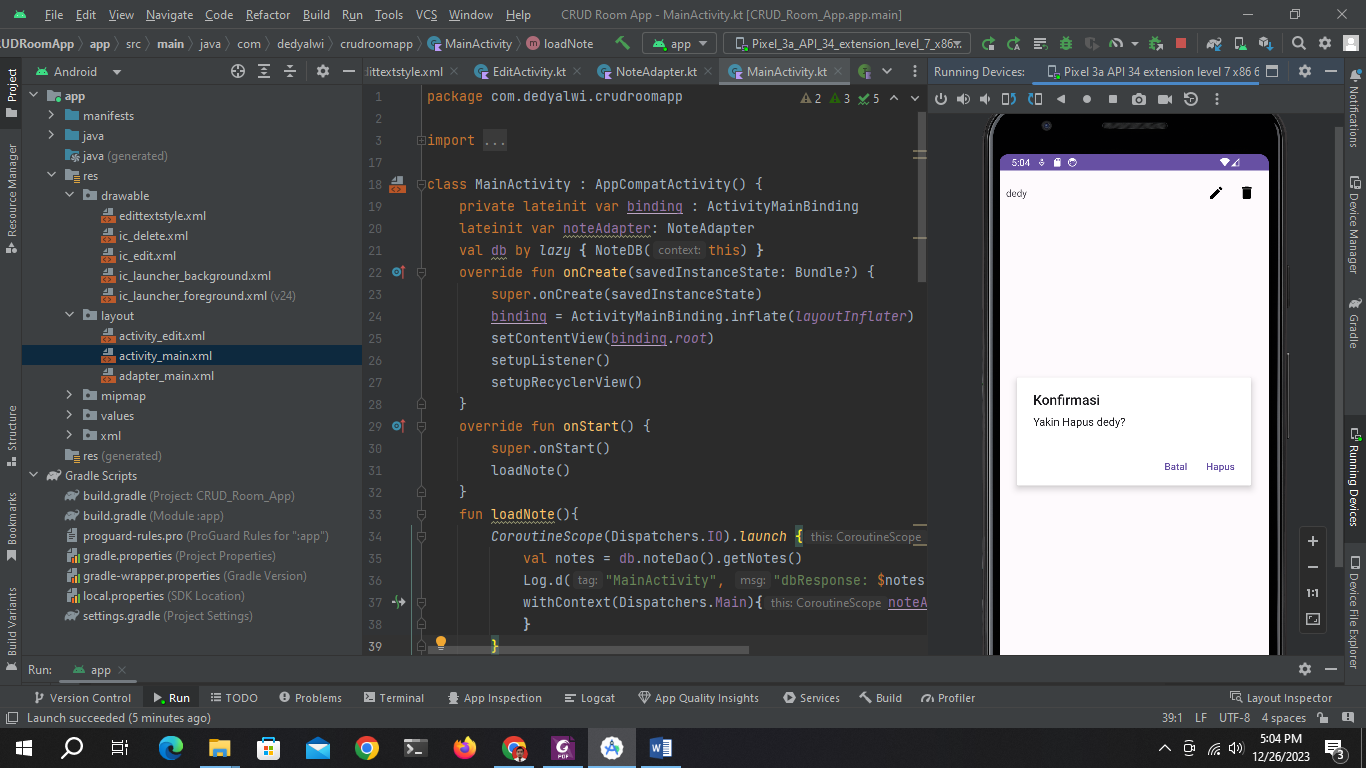
Project : Aplikasi CRUD Daftar Buku











package com.dedyalwi.crudroomapp  
  
import android.content.Intent  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.util.Log  
import androidx.appcompat.app.AlertDialog  
import androidx.recyclerview.widget.LinearLayoutManager  
import com.dedyalwi.crudroomapp.databinding.ActivityMainBinding  
import com.dedyalwi.crudroomapp.room.Constant  
import com.dedyalwi.crudroomapp.room.Note  
import com.dedyalwi.crudroomapp.room.NoteDB  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.withContext  
  
class MainActivity : AppCompatActivity() {  
 private lateinit var binding : ActivityMainBinding  
 lateinit var noteAdapter: NoteAdapter  
 val db by *lazy* **{** NoteDB(this) **}** override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 binding = ActivityMainBinding.inflate(*layoutInflater*)  
 setContentView(binding.*root*)  
 setupListener()  
 setupRecyclerView()  
 }  
 override fun onStart() {  
 super.onStart()  
 loadNote()  
 }  
 fun loadNote(){  
 *CoroutineScope*(Dispatchers.IO).*launch* **{** val notes = db.noteDao().getNotes()  
 Log.d("MainActivity", "dbResponse: $notes")  
 withContext(Dispatchers.Main)**{**noteAdapter.setData(notes)  
 **}  
 }** }  
 private fun setupListener() {  
 binding.buttonCreate.setOnClickListener **{**//startActivity(Intent(this, EditActivity::class.java))  
 intentEdit(0, Constant.TYPE\_CREATE)  
 **}** }  
 fun intentEdit(noteId: Int, intentType: Int){  
 startActivity(  
 Intent(*applicationContext*, EditActivity::class.*java*)  
 .putExtra("intent\_id", noteId)  
 .putExtra("intent\_type", intentType)  
 )  
 }  
 private fun setupRecyclerView() {  
 noteAdapter = NoteAdapter(*arrayListOf*(), object : NoteAdapter.OnAdapterListener{  
 override fun onRead(note: Note) {  
 intentEdit(note.id, Constant.TYPE\_READ)  
 }  
 override fun onUpdate(note: Note) {  
 intentEdit(note.id, Constant.TYPE\_UPDATE)  
 }  
 override fun onDelete(note: Note) {  
 deleteDialog(note)  
 }  
 })  
 binding.listNote.*apply* **{** *layoutManager* = LinearLayoutManager(*applicationContext*)  
 *adapter* = noteAdapter  
 **}** }  
 private fun deleteDialog(note: Note){  
 val alertDialog = AlertDialog.Builder(this)  
 alertDialog.*apply* **{** setTitle("Konfirmasi")  
 setMessage("Yakin Hapus ${note.title}?" )  
 setNegativeButton("Batal") **{** dialogInterface, i **->** dialogInterface.dismiss()  
 **}** setPositiveButton("Hapus") **{** dialogInterface, i **->** dialogInterface.dismiss()  
 *CoroutineScope*(Dispatchers.IO).*launch* **{** db.noteDao().deleteNote(note)  
 loadNote()  
 **}  
 }  
 }** alertDialog.show()  
 }  
}

package com.dedyalwi.crudroomapp  
  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.view.View  
import com.dedyalwi.crudroomapp.databinding.ActivityEditBinding  
import com.dedyalwi.crudroomapp.room.Constant  
import com.dedyalwi.crudroomapp.room.Note  
import com.dedyalwi.crudroomapp.room.NoteDB  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
  
class EditActivity : AppCompatActivity() {  
 val db by *lazy* **{** NoteDB(this) **}** private var noteId: Int = 0  
 private lateinit var binding : ActivityEditBinding  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 binding = ActivityEditBinding.inflate(*layoutInflater*)  
 setContentView(binding.*root*)  
 setupView()  
 setupListener()  
 }  
 fun setupView(){  
 val intentType = *intent*.getIntExtra("intent\_type", 0)  
 when(intentType){  
 Constant.TYPE\_CREATE -> {  
 binding.buttonUpdate.*visibility* = View.*GONE* }  
 Constant.TYPE\_READ -> {  
 binding.buttonSave.*visibility* = View.*GONE* binding.buttonUpdate.*visibility* = View.*GONE* getNote()  
 }  
 Constant.TYPE\_UPDATE -> {  
 binding.buttonSave.*visibility* = View.*GONE* getNote()  
 }  
 }  
 }  
 private fun setupListener() {  
 binding.buttonSave.setOnClickListener **{** *CoroutineScope*(Dispatchers.IO).*launch* **{** db.noteDao().addNote(  
 Note(0, binding.editTitle.*text*.toString(),  
 binding.editNote.*text*.toString())  
 )  
 finish()  
 **}  
 }** binding.buttonUpdate.setOnClickListener **{** *CoroutineScope*(Dispatchers.IO).*launch* **{** db.noteDao().updateNote(  
 Note(noteId, binding.editTitle.*text*.toString(),  
 binding.editNote.*text*.toString())  
 )  
 finish()**}  
 }** }  
 fun getNote(){  
 noteId = *intent*.getIntExtra("intent\_id", 0)  
 *CoroutineScope*(Dispatchers.IO).*launch* **{** val notes = db.noteDao().getNote(noteId)[0]  
 binding.editTitle.setText(notes.title)  
 binding.editNote.setText(notes.note)  
 **}** }  
 override fun onSupportNavigateUp(): Boolean {  
 onBackPressed()  
 return super.onSupportNavigateUp()  
 }  
}

package com.dedyalwi.crudroomapp  
  
import android.view.LayoutInflater  
import android.view.ViewGroup  
import androidx.recyclerview.widget.RecyclerView  
import com.dedyalwi.crudroomapp.databinding.AdapterMainBinding  
import com.dedyalwi.crudroomapp.room.Note  
  
class NoteAdapter (private var notes: ArrayList<Note>, private val listener:  
OnAdapterListener) :  
 RecyclerView.Adapter<NoteAdapter.NoteViewHolder>(){  
 inner class NoteViewHolder(val binding: AdapterMainBinding) :  
 RecyclerView.ViewHolder(binding.*root*)  
 interface OnAdapterListener {  
 fun onRead(note: Note)  
 fun onUpdate(note: Note)  
 fun onDelete(note: Note)  
 }private lateinit var binding : AdapterMainBinding  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): NoteViewHolder {  
 val binding = AdapterMainBinding.inflate(  
 LayoutInflater.from(parent.*context*),  
 parent, false)  
 return NoteViewHolder(binding)  
 }  
 override fun onBindViewHolder(holder: NoteViewHolder, position: Int) {  
 *with*(holder) **{** *with*(notes[position]) **{** binding.textTitle.*text* = title  
 binding.textTitle.setOnClickListener **{** listener.onRead(notes[position])  
 **}** binding.iconEdit.setOnClickListener **{** listener.onUpdate(notes[position])  
 **}** binding.iconDelete.setOnClickListener **{** listener.onDelete(notes[position])  
 **}  
 }  
 }** }  
 override fun getItemCount() = notes.size  
 fun setData(newList: List<Note>) {  
 notes.clear()  
 notes.addAll(newList)  
 notifyDataSetChanged()  
 }  
}

package com.dedyalwi.crudroomapp.room  
  
import android.content.Context  
import androidx.room.Database  
import androidx.room.Room  
import androidx.room.RoomDatabase  
  
@Database(  
 entities = [Note::class],  
 version = 1  
)  
abstract class NoteDB: RoomDatabase() {  
 abstract fun noteDao(): NoteDao  
 companion object {  
 @Volatile private var instance: NoteDB? = null  
 private val LOCK = Any()  
 operator fun invoke(context: Context) = instance ?: *synchronized*(LOCK)**{** instance ?: buildDatabase(context).*also***{** instance = **it}  
 }** private fun buildDatabase(context: Context)= Room.databaseBuilder(  
 context.*applicationContext*,  
 NoteDB::class.*java*,  
 "note12345.db"  
 ).build()  
 }  
}

package com.dedyalwi.crudroomapp.room  
  
import androidx.room.Dao  
import androidx.room.Delete  
import androidx.room.Insert  
import androidx.room.Query  
import androidx.room.Update  
  
@Dao  
interface NoteDao {  
 @Insert  
 fun addNote(note: Note)  
 @Update  
 fun updateNote(note: Note)  
 @Delete  
 fun deleteNote(note: Note)  
 @Query("SELECT \* FROM note")  
 fun getNotes(): List<Note>  
 @Query("SELECT \* FROM note WHERE id=:note\_id")  
 fun getNote(note\_id: Int): List<Note>  
}

<?xml version="1.0" encoding="utf-8"?>  
<shape xmlns:android="http://schemas.android.com/apk/res/android">  
 <corners android:radius="10dp" />  
 <padding android:bottom="15dp"  
 android:right="15dp"  
 android:left="15dp"  
 android:top="15dp"/>  
 <solid android:color="#f3f6f4"/>  
</shape>

package com.dedyalwi.crudroomapp.room  
  
class Constant {  
 companion object {  
 const val TYPE\_READ = 0  
 const val TYPE\_CREATE = 1  
 const val TYPE\_UPDATE = 2  
 }  
}